Attorney Docket No.: 032374-003

WEB-BASED MEDIA SUBMISSION TOOL

The present application is related by subject matter to U.S. Application Serial No. 09/440,461 ~~(Atty. Dkt. No. 032374-002) entitled~~ now Pat. No. 6,732,162. ~~_____, filed on even date herewith and incorporated herein by reference.~~

TH 11/5/04

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to the handling, manipulation and processing of digital content and more particularly to the transportation and Internet publishing of digital content, particularly image media objects and rich media.

2. State of the Art

Much of the phenomenal success of the web is attributable to its graphical nature. Literally, a picture is worth a thousand words. The capture of digital images has become routine, using digital cameras and scanners. Nevertheless, although the handling of images by web-site creators has achieved a high degree of automation, for the average technology user (the "imaging civilian"), manipulating and sharing digital images over the Internet remains a cumbersome and daunting process. Piecemeal solutions that have been devised for handling digital images require a level of sophistication that is beyond that of the ordinary user. For example, transferring a digital image may require first downloading a FTP program, then installing it, then running it and connectting to an FTP server by typing the server name in the connection dialog, then navigating to the proper subdirectory, selecting the files to be uploaded, making sure that the program is in binary transfer mode, then sending the files. For the imaging civilian, such an

involved process can be daunting to say the least.

Additionally, as technologies advance and casual users begin to experiment with other media objects, such as streaming video, 3D objects, slide shows, graphics, movies, and even sound files that accompany imaging data, the processes required to share these rich media types on the Internet becomes exponentially more complicated and prohibitive. As the realization of the Internet as an interactive, content rich medium becomes more and more a reality, the need for enabling the use and distribution of rich content and media on the Internet will become the gating factor to its long term success.

A broad-based solution to the foregoing problem requires a web-based media submission tool that allows for submission of media objects in a convenient, intuitive manner. A company named Caught in the Web, has attempted to create a broad-based media submission tool known as "ActiveUpload". ActiveUpload allows an arbitrary file to be dragged and dropped onto a web page control for upload to the web server. An ActiveUpload control allows users to, without leaving a web page, transfer files to a server (Internet or intranet) by selecting the files on the user's desktop that the user wants to transfer, then dragging them onto the web page. For example, a user, having visited a web page, can contribute pictures, documents, zip files, etc., without having to leave the web page and use an FTP program. Standard web authoring tools can be used to integrate ActiveUpload into web pages and change the behavior of the control.

Although Caught in the Web's ActiveUpload tool simplifies the user experience, it does little toward furthering "backend" automation in the handling and distribution of media objects and has no built in "intelligence" to streamline the process of handling and transporting rich media objects from the front end.

SUMMARY OF THE INVENTION

The present invention, generally speaking, provides an improved web-based media submission tool. As with some existing tools, operation of the tool is drag and drop or the user can "click" to browse a directory to select media objects. Unlike existing tools, the tool provides several unique and valuable functions. For example, the tool provides the user an opportunity to confirm the submission with a visual representation, for example by generating a thumbnail image of the rich media file that has been selected. Additionally, batch submission is provided to allow a user to drag and drop or select a plurality of images or other media objects. Submission from a web page to a web page is also provided for. Even more importantly, the submission tool is configurable to perform a variable amount of intelligent preprocessing on media objects prior to upload. In the case of digital images, the tool can perform sizing and formatting, for example. Information capture is performed with information being uploaded together with the media objects. In an exemplary embodiment, information capture is both user-transparent (e.g., user ID and/or password) and user-visible (e.g., the user can provide captions for media objects). The submission of information about the user and the media objects facilitates automatic integration of the media objects within existing databases.

BRIEF DESCRIPTION OF THE DRAWING

The present invention may be further understood from the following description in conjunction with the appended drawing. In the drawing:

Fig. 1 is a diagram of an exemplary web page providing media object acquisition functions;

Fig. 2 is a diagram of another exemplary web page providing image acquisition functions;

Fig. 3 is a table pertaining to a first portion of the Prepare and Post component design; and

Fig. 4 is a table pertaining to a second portion of the Prepare and Post component design.

DETAILED DESCRIPTION OF THE PRFERRED EMBODIMENTS

The following describes the Prepare and Post™ tools, which prepares and submits media objects from inside a standard browser, referred to as the first location, to a second location or server. The media objects may be pictures (images), movies, videos, graphics, sound clips, etc. Although in the following description the submission of images is described in greatest detail, the same principles apply equally to media obejcts of all descriptions.

The Prepare and Post tools refers to browser-side components which together provide the ability to submit and transport media objects over the web to be stored and served. Using the Prepare and Post tools, end users can submit images in an immediate, intuitive manner. No technical sophistication is required. In particular, understanding technical terms such as JPEG, resolution, pixel, kilobyte, transfer protocol, IP address, FTP etc., is not required, since the Prepare and Post tools handles all of these tasks for the user. The benefits of the Prepare and Post tool are:

a) to the image submitter, the ability to submit media objects to web pages immediately without needing to overcome technical obstacles;

b) to the image submitter, the ability to submit media objects to web pages "as is" without making modifications to the media objects prior to sending.

c) to PictureWorks web site partner, access to a uniform, standardized, reliable and secure channel for media acquisition;

d) to PictureWorks web site partner, access to contributed media "made to order", it meets their imaging specifications every time without human intervention;

e) to PictureWorks web site partner, the ability to provide web site visitors with an easy, error free way to contribute media;

f) to PictureWorks web site partner, access to contributed media in "real time" with no time delays.

The two primary components used in the Prepare and Post tools which carry out these functions are 1) the media object identifier and 2) the media sender.

In general, the media object identifier functions to provide a graphical interface for placing and associating a media object from a user's desktop onto a web page. The media sender carriers out the function of transmitting media objects to a second location.

There are two ways media objects on the first location become associated with a media object identifier. The first is through a "drag and drop" behavior where the user clicks on a media object to select the one they want to submit. The media object is then dragged to the media object identifier. Releasing the mouse button associates the media object with the media object identifier. This behavior is allowed in web browsers that support drag and drop functionality. The Prepare and Post tools enable these browsers to accept media objects via drag and drop by providing the media object identifier as an ActiveX component.

The second way to associate a media object on the first location with the media object identifier is to click on the media object identifier to browse for media objects, then select the media object of choice. This method is made available for web browsers where the media object identifier needs to be a pure Java component. (Such "signed applet browers" like Netscape Navigator) In this instance, the user may be asked to choose a media object in a similar manner as

when choosing a file to be opened, either by graphical navigation or by specifying a path name. For example, a prompt associated with the media object identifier may be displayed prompting the user to click within the media object identifier. Clicking within the media object identifier brings up a browse dialog. Using the browse dialog, the user selects the desired media object, which is then placed in the media object identifier. The Prepare and Post tools will generate a visual representation or thumbnail of the media object, a feature currently not available in signed applet browsers.

Real estate is an example of a prime application of the Prepare and Post tools. "Curb appeal" is of great importance in the realty industry and can only be judged by "drive-bys," which are time-consuming and laborious, or by the availability of images. The Prepare and Post tools make real estate images readily available with a minimal amount of effort.

Referring to Figure 1, an example is shown of a realty web page featuring the described Prepare and Post tools functionality. The user associates images with a media object identifier via the methods described above and selects appropriate captions for the images, e.g., living room, family room, etc. The captions may be typed in or selected from menus. The user also supplies identifying information, in this instance the MLS listing number. When the user clicks the Send button, the images are uploaded and processed immediately according to the configuration of the Prepare and Post tools.

The Prepare and Post tools also support a batch interface, allowing a plurality of images to be submitted simultaneously as in the case of a professional photographer, for example. The opportunity for user confirmation is again provided, e.g., by displaying a visual representation of the images in the batch.

If a mistake is made such that the wrong image is placed in an media object identifier, the correct image may be placed in the media object identifier. The correct image will replace the mistaken image. Alternatively, the user may remove an image from a media object identifier by right-clicking on the media object identifier and selecting Remove within a resulting pop-up menu.

Note that any number of media object identifiers may be provided on a web page and that the media object identifiers may be separate or grouped. This is evident in Figure 2. The number of media object identifiers provided on a page can be pre-configured and fixed, allowing no user intervention, or the media object identifiers can be generated dynamically, allowing the user to determine how many media object identifiers they need for media submission. Figure 2 shows a web page with various sizes of media object identifiers. If a media object identifier is separate, its image will be transmitted separately to the second location. If an media object identifier is part of a group, its image will be transmitted to the second location as part of a group of images that are stored together and cataloged together. Media object identifiers that are associated together as a group are noted as such in the web page interface and transparently in the media object identifier object code. Moreover, a web page may have multiple groups of media object identifiers, or "groups of groups."

The usefulness of images is greatly enhanced by capturing and identifying information about the images and submitting the identifying information with the images. Information may be image-specific, user-specific or both. The submission of information about the user and the media objects facilitates automatic integration of the media objects within existing databases. Information capture may be overt or covert or both. This unique automatic database integration enables the images to be served with the proper web page data. Overt information capture relies upon the user to make menu selections of appropriate captions as illustrated in Figure 1, or to make text entries within text fields, or both. The Prepare and Post tools are easily customized to accept menu selections and text fields for different applications. Covert information capture occurs by having the web browser automatically pass to the Prepare and Post tools known information such as a user ID or password used to access the web page.

A key differentiator of the Prepare and Post tools is the browser, or client-side intelligence built into the tools. This intelligence directly provides features including those already outlined such as associating data with media objects, generating a visual representation of the media objects and generating media

object identifiers dynamically or in a pre-set manner. Other features are also provided via this intelligence, specifically, the ability to control the width and height of the media object identifier and the ability to preprocess the media objects in any number of ways prior to transporting to a second location. In the case of an image media object for example, the Prepare and Post tools may resize the image, (i.e., increase or decrease its size as defined by either physical dimensions, pixel count, or kilobytes). Compression, for example, is a type of sizing. The Prepare and Post tools may also change the image's file format (a way of a media object being identified as to a "type" or "kind" of media), change the quality setting of the image, crop the image or change the aspect ratio, add text or annotations, encode or combine (including stitching) the media object, or enhance the media object by changing image values, for example, relating to contrast or saturation. This intelligence may be executed in a manner that is transparent to the end user. This transparency allows the end user to submit media to the Prepare and Post tools "as is," since the tools will automatically prepare it to meet the requirements of the second location. Note that, although image submission may involve client-side processing, image processing is not required.

The Prepare and Post tools are available for customers to integrate into their own web pages. The Prepare and Post tools are easily integrated into web sites (customers) to allows those sites to accept media objects from web site visitors (users). Appendix A is a generic HTML HostTemplate illustrating how Prepare and Post components are integrated into a web page. The HTML template file (which is a complete working example) contains instructions and a few small code snippets that the customer pastes into the web page. Integrating the Prepare and Post components requires an Initialization Section, a Configuration Section, an ImageWell (media object identifier) Section, a Submission Section and an ImageUpLoad Control Section. To include the Prepare and Post tools media object identifiers on a web page, the customer cuts and pastes code snippets for these sections from the template into the web page.

The Initialization Section consists of a few lines of JavaScript code that will download all of the needed Prepare and Post submission components.

The Configuration Section overrides various configurable default settings that the customer can control. In the Configuration Section, the media object identifier component is sized and configured to perform any preprocessing of the image that may be desired prior to upload. Configurable parameters include both fixed values for all submissions (per submission values) and fixed values for all images within a submission (per image values), as will be explained presently.

Fixed values for all submissions include *DefaultImageWidth* and *DefaultImageHeight*, as well as include *DefaultControlWidth* and *DefaultControlHeight.* The former specify the default width and height of the images after they have been compressed for transmission. The latter specify the default width and height of all media object identifiers. To create media object identifiers having different sizes, the customer specifies the desired size when creating the media object identifier. Another fixed value for all submissions is *Quality*. This determines the quality level of the images after they have been compressed for transmission (0 is the lowest quality/highest compression and 100 is the highest quality/lowest compression).

Fixed values for all media objects within a submission include *Key1* and *Key2*. *Key 1* is the primary value that determines the filename of the resulting image file and, consequently, its URL. It is important that each submitted image have a unique name to prevent one image from overwriting another. *Key2* is an optional secondary key that is appended to *Key1* before the image's filename and URL are created. While default values for *Key1* and *Key2* can be specified in the configuration section, more likely this value will be supplied from a field in the web form. If the web page form contains a control named "Key1," then its value will be used for this key. For example, the field Key1 might be labeled as "MLS Number" on the web page. Similarly, the field Key2 might be labeled "Zip Code" on the web page. A sequence number is appended to the Key1/Key2 combination. When there are multiple media object identifiers on a page, this will ensure that each image has a unique key.

All media object identifiers on a web page must be contained within an

HTML form. A single line of JavaScript code is inserted into the web page (within the HTML form) in each place where a media object identifier is desired. The Media object identifier Section can specify the width and height for each media object identifier. If the width and height are omitted, then the default width and heigth from the Configuration Section are used.

The Submission Code Section contains HTML code that creates the button that submits both the images to the second locations and the form to the customer's server. Within the Submission Code Section, an HTML "href" parameter is required for the Send Button that causes the images to be sent. After the images have been sent, the web page form will be submitted in the standard manner. The form must define two hidden fields named "url" and imagecount." The imagecount field will contain the number of images actually transmitted. In an exemplary embodiment, the URL for images 2 through "n" are generated by replacing the initial sequence number at the end of the returned URL with the desired image number.

The ImageUpload Control Section holds a small piece of JavaScript code that is placed at the very end of the body section of the web page. This code creates the non-visible Image Upload control, or media sender, that performs the transfer of images from the user's machine to the second location.

The Prepare and Post components support multiple browsers and dynamically adjust their behavior according to the type of browser that is currently running. For example, under supported versions of Microsoft's browsers, media object identifiers are implemented as ActiveX controls, while under supported Netscape browsers, media object identifiers are implemented as Java Applets. This multiple browser support is completely automatic.

Figures 3 and 4 present further details of the media object identifier and media sender components, respectively.

From the foregoing description, it will be appreciated that the present media submission tool, besides offering convenience to the end user, offers

convenience and flexibility to technology partners. In particular, web page integration is designed to facilitate automatic server-side integration of media content.

It will be apparent to those of ordinary skill in the art that the present invention can be embodied in other specific forms without departing from the spirit or essential character thereof. The presently disclosed embodiments are therefore considered in all respects to be illustrative and not restrictive. The scope of the invention is indicated by the appended claims rather than the foregoing description, and all changes which come within the meaning and range of equivalents thereof are intended to be embraced therein.

APPENDIX A: HostTemplate generic.htm

```
<HTML>
<HEAD>

<!--********************************************** Begin Initialization Section -->
<!--***** This section of code must appear at  -->
<!--***** the beginng of the <HEAD> section of -->
<!--***** your web page. Copy this code and    -->
<!--***** paste it directly into your web page.-->
<SCRIPT type="text/javascript" src="http://157.22.134.49/company/pwtcomponents.js"></SCR
IPT>
<SCRIPT type="text/javascript" src="http://157.22.134.49/company/company.js"></SCRIPT>
<!--********************************************** End Initialization Section -->

</HEAD>
<BODY>

<!--********************************************** Begin Configuration Section -->
<!--***** This section of code must appear     -->
<!--***** anywhere after the initialization    -->
<!--***** section (above), and before the      -->
<!--***** the <FORM> that contains the image   -->
<!--***** wells.                               -->
<!--*****                                      -->
<!--***** This section defines data values     -->
<!--***** needed by the image wells. You can   -->
<!--***** modify these values to suit  -->
<!--***** your needs.                          -->
<SCRIPT Language="Javascript">
PWT.Key1 = "name-your-image-here";  // If the <FORM> contains fields named 'Key1'
PWT.Key2 = " ";                     //   & 'Key2' their values will be used.
PWT.Quality = 93;
PWT.DefaultImageWidth = 640;
PWT.DefaultImageHeight = 480;
PWT.DefaultControlWidth = 326;      // Includes a 3 pixel border
PWT.DefaultControlHeight = 246;         // Include a 3 pixel border
</SCRIPT>
<!--********************************************** End Configuration Section -->
```

```
<FORM>
This sample displays a working image well.
<BR>

<!--********************************************** Begin ImageWell Section -->
<!--***** This code creates an image well on    -->
<!--***** the web page. While this template     -->
<!--***** only contains a single image well,    -->
<!--***** you can use as many as you like.      -->
<!--***** Copy this code into your web page     -->
<!--***** anywhere within your <FORM> where     -->
<!--***** you want an image well to appear.     -->
<SCRIPT Language="Javascript">
PWT.addimagecontrol();      // or "PWT.addimagecontrol(640,480);" to override
                            // the default width and height.
</SCRIPT>
<!--********************************************** End ImageWell Section -->

<BR>
This text is after the image well.


<P>
<!--********************************************** Begin Submission Code Section -->
<!--***** You can use any type of button you    -->
<!--***** wish, but rather than it being a      -->
<!--***** standard SUBMIT button, it must       -->
<!--***** instead contain the parameter:        -->
<!--*****                                       -->
<!--*****       onclick="PWT.Submit()"          -->
<!--*****                                       -->
<!--***** (as shown in the example below).      -->
<!--***** After the images have been sent,      -->
<!--***** your web page FORM will be submitted  -->
<!--***** in the standard manner.               -->
<!--*****                                       -->
```

```
<!--***** Your FORM must define two hidden     -->
<!--***** fields named "url" & "imagecount"    -->
<!--***** (see examples below). The "url"      -->
<!--***** field will be populated with the     -->
<!--***** resulting URL of the first (or only) -->
<!--***** image submitted, and the "imagecount"-->
<!--***** field will contain the number  of    -->
<!--***** images actually transmitted. The URL -->
<!--***** for images 2 thru n can be generated -->
<!--***** by replacing the initial sequence
<!--***** number (which will always be "1")    -->
<!--***** at the end of the returned URL with  -->
->
<!--***** the desired image number.            -->
<INPUT type="hidden" name="url">
<INPUT type="hidden" name="imagecount">
<INPUT type="button" value="Submit Images" onclick="PWT.Submit()">
</FORM>
<!--********************************************** End Submission Code Section -->

<!--********************************************** Begin ImageUpload Control Section -->
<!--***** This section of code must appear at  -->
<!--***** the end of the <BODY> section of     -->
<!--***** your web page. Copy this code and    -->
<!--***** paste it directly into your web page.-->
<SCRIPT Language="Javascript">
PWT.adduploadcontrol();
</SCRIPT>
<!--********************************************** End ImageUpload Control Section -->
</BODY>
</HTML>
```